SISTEMI OPERATIVI

ESERCIZIO N. 1 del 30 GIUGNO 2000

Il dipartimento di applicazioni multimediali di una società informatica è composto da **N** workstation e utilizza uno **storage system** centralizzato per l'immagazzinamento dei file multimediali e per il backup dei dischi delle workstation. Tre tipi di processi possono accedere concorrentemente dalle workstation allo storage system: **processi M**, che *leggono* e *scrivono* file multimediali, **processi B** che effettuano il backup e **processi R** che ripristinano i dati di backup. La politica di accesso impone che processi di tipo **M** non possano accedere contemporaneamente a processi di tipo **B**, e che *dalla stessa workstation* non possano accedere contemporaneamente processi **R** e **B**.
Si implementi una soluzione usando il costrutto monitor per modellare lo storage system e i processi M, R e B, e si descriva la sincronizzazione tra i processi. Nella soluzione si dia priorità all'utilizzo delle risorse. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare starvation.

Nota: in questa soluzione si suppone che ci siano un solo processo B e un solo processo R da ogni workstation.

```
program dipartimento_multimediale

const N = ...; { numero di workstation }

type   tipo = (M, B, R);
       workstation = 1..N

type processo = process(t: tipo, w: workstation)
begin
      storage.accedi(t, w);
      <esegui il compito>
      storage.rilascia(t, w);
end

type storage_system = monitor

{ variabili del monitor }
var  numero: array [tipo] of integer;
      { numero di processi per tipo che stanno usando lo
      storage }
      incoda: array [tipo] of integer;
      { numero di processi in coda per tipo }
      occupato: array [workstation] of boolean;
      { dice se c'è già un processo R o B da una data
      workstation }
      coda: array [tipo] of condition;
      { code su cui sospendere i processi }
```

```
procedure entry accedi(t: tipo; w: workstation)
begin
  if t = M then
    { multimedia: non ci devono essere backup }
    while numero[B] > 0 do
      begin incoda[M]++; coda[M].wait; incoda[M]--; end
  else if t = R then
    { restore: non ci deve essere il corrispondente backup }
    while occupato[w] do
      begin incoda[R]++; coda[R].wait; incoda[R]--; end
 else {caso t = B }
    { backup: no multimedia né il corrispondente restore }
    while numero[M] > 0 or occupato[w] do
      begin incoda[B]++; coda[B].wait; incoda[B]--; end

  numero[t] := numero[t] + 1;
  if t = R or t = B then
    occupato[w] := true;
end
```

```
procedure entry rilascia(t: tipo; w: workstation)
var  s: integer;
begin
  if t = R or t = B then
    occupato[w] := false;
 numero[t] := numero[t] – 1;

  if t = M and numero[M] = 0 then
  begin
      s := incoda[B];
      for i := 1 to s do        coda[B].signal;
  end

  if t = B then
  begin
    if numero[B] = 0 then
    begin
      s := incoda[M];
      for i := 1 to s do        coda[M].signal;
    end
    s := incoda[R];
      for i := 1 to s do        coda[R].signal;
  end

if t = R then
begin { devo risvegliarli tutti, ma al massimo uno esegue }
  s := incoda[B];
  for i := 1 to s do      coda[B].signal;
end
end
```

```
begin { inizializzazione delle variabili }
        var k: tipo; i: workstation;
        for k := M to R do
                numero[k] := 0;
        for i := 1 to N do
                occupato[i] := false;
end

var  storage: storage_system; { il nostro monitor }
        pm1, pm2, ... : processo (M, k);
        pb1, pb2, ... : processo (B, j);
        pr1, pr2, ... : processo (R, l);

begin end.
```

Considerazioni

La sospensione è fatta con un *while* per tutti i tipi di processo. In realtà, per i processi M sarebbe stato sufficiente un *if*.
I processi B che escono, risvegliano tutti i processi R in coda, non sapendo se c'è e quale è il processo R della stessa workstation. Stessa cosa per i processi R che escono.

Nel caso in cui ci siano più processi R e B da ogni workstation, bisogna tenerne conto usando dei contatori invece che dei valori booleani (variabile occupato).

Nella soluzione proposta si è fatto uso di un array di code. Altre soluzioni potrebbero contemplare una coda unica o una matrice di code. In questo ultimo caso, il risveglio può essere fatto solo sui processi B o R che effettivamente possono eseguire.

Starvation

La soluzione proposta presenta starvation, infatti è possibile che i processi di tipo M monopolizzino lo storage system e non permettano ai processi di tipo B di entrare, e viceversa.

Un modo per evitare starvation è quello di utilizzare dei contatori, in modo che dopo E esecuzioni dei processi M/B, si lasci l'accesso ai processi B/M.